COBB

INSIDE DOS

Tips & techniques for MS-DOS & PC-DOS

April 1992
Vol. 3 No. 4

Increase your DOS command-line power with pipe filters

VERSIONS 2.1-5.0

DOS provides several powerful tools that let you redirect command output and input. By using <, >, >>, and |, you can send information to a DOS command from a text file, send the output of a DOS command to a text file or a printer, or append the output of a command to an existing text file. The final redirection symbol, the pipe (|), allows you to customize the output of a DOS command by passing it through the MORE, FIND, and SORT filter commands.

Using the pipe in combination with MORE, FIND, and SORT substantially increases your power from the DOS command line. You have the flexibility to obtain quickly only the information you want from a command's output. At the same time, you can organize that output into a number of different formats. By using the pipe in combination with the other redirection symbols, you can generate, process, and store the output of a DOS command in a single statement.

How the pipe works

The idea behind the pipe is simple. A DOS command produces a set of output information. For example, the DIR command produces a list of all the files in your directory. The pipe symbol takes the output of a command and processes it using one or more of the three DOS filters: MORE, SORT, and FIND. On the screen, you see the output after the filter finishes processing it.

The pipe command works much like the switches you add to certain DOS commands. You're probably familiar with the /P and /W switches available with the DIR command. The /P switch allows you to see the output of a long directory in a page-by-page format, and the /W switch lets you show the filenames in a row-and-column layout. The switches take the normal output, a long list of file names, and customize it.

In the same way, the pipe takes command output and passes it through a filter which processes the information and then displays the result on the screen. Some commands, like DIR, offer switches that allow you to filter the output without using the redirection symbols. Like the switches, when you send the output of commands like TYPE through the MORE or SORT filter, you'll get more meaningful data that's easier to read and interpret.

Before you get started, however, you need to check the DOS environment settings. The pipe uses the TEMP environment variable. If it's not defined, the pipe won't work. Fortunately, working with TEMP is really easy.

First check this value by typing in SET at the DOS command prompt. You'll see a list similar to the one shown below:

```
COMSPEC = C:\COMMAND.COM
PROMPT = $p$g
PATH = C:\DOS;C:\WINDOWS
```

If TEMP isn't in your list, add it by typing

```
C:\>set temp C:\DOS
```

DOS will add the line

```
TEMP = C:\DOS
```

to your environment. If you're using a system without a hard disk, set the TEMP file to A: instead of C:\DOS. Now you're ready to go with the pipe.

Let's take a look at how the pipe filters your command output. Along the way we'll provide a few useful commands using the pipe to give you more flexibility from the DOS command line.

Continued on page 10

IN THIS ISSUE

INSIDE DOS™

Inside DOS (ISSN 1049-5320) is published monthly by The Cobb Group.

Prices	Domestic	$59/yr. ($7.00 each)
	Outside the US	$79/yr. ($8.50 each)
Address	The Cobb Group 9420 Bunsen Parkway, Suite 300 Louisville, KY 40220	
Phone	Toll free	(800) 223-8720
	Local	(502) 491-1900
	FAX	(502) 491-8050
Staff	Editor-in-Chief	Jim Welp
	Consulting Editors	Mark W. Crane
		Tim Landgrave
	Contributing Editor	Van Wolverton
	Assistant Editor	Michael Stephens
	Editing	Clyde Zellers
		Polly Blakemore
		Cecilia Crosby-Lampkin
	Production	Debbie Lane
	Design	Karl Feige
	Publications Manager	Elayne Noltemeyer
	Publisher	Douglas Cobb

Address correspondence and special requests to The Editor, *Inside DOS*, 9420 Bunsen Parkway, Suite 300, Louisville, KY 40220. Address subscriptions, fulfillment questions, and requests for bulk orders to Customer Relations, at the same address.

Postmaster: Send address changes to *Inside DOS*, P.O. Box 35160, Louisville, KY 40232. Second class postage is paid in Louisville, KY.

Conventions

To avoid confusion, we would like to explain a few of the conventions used in *Inside DOS*.

When we instruct you to type something, those characters usually appear on a separate line along with the DOS prompt. The characters you type will appear in red, while the characters DOS displays will appear in black.

Occasionally, we won't display the command you type on a separate line. In these cases, we'll display the characters you type in italics. For example, we might say, "Issue the command *dir *.txt* at the DOS prompt." Although DOS is not case-sensitive, we'll always display the characters you type in lowercase.

When we refer to a general DOS command (not the command you actually type at the DOS prompt), we'll display that command name in all caps. For example, we might say, "You can use either the COPY or XCOPY command to transfer files from one disk to another."

Many commands accept parameters that specify a particular file, disk drive, or other option. When we show you the form of such a command, its parameters will appear in italics. For example, the form of the COPY command is

copy *file1 file2*

where *file1* and *file2* represent the names of the source file and the target file, respectively.

The names of keys, such as [Shift], [Ctrl], and [F1], appear in brackets. When two keys must be pressed simultaneously, those key names appear side by side, as in [Ctrl][Break] or [Ctrl]z.

LETTERS

Mysterious batch files

VERSION 5.0

On the front page of the January 1992 issue of *Inside DOS*, you showed a DOS 5 Shell window that displayed the contents of the \DOS directory. I noticed in that figure some weird filenames of 8 bytes each, such as

```
1046DOSC.BAT          8        05-14-91
```

I now have 12 such files of either 8 bytes or 32 bytes each. Five of these files contain

```
@COMMAND
```

Six files contain

```
@QBASIC \run C:\DOS\REMLINE.BAS
```

and one file contains

```
@QBASIC \run C:\DOS\GORILLA.BAS
```

These 12 files were not in the \DOS directory when I bought my first and only computer three months ago. I feel that somehow they were created by QBASIC. I have played the "Gorilla" and "Nibbles" games several times by clicking on the filenames in the Shell. Are these small files necessary? They're repetitious and appear to me to be junk. Can I delete them?

A. C. Atkins
Fort Stocton, Texas

The mysterious batch files are temporary files DOS creates when you run a program or a batch file from the DOS Shell. Every time you run a program or a batch file from the Shell, DOS creates a temporary batch file with a unique name that either runs the command interpreter or loads the program.

If you run a program, DOS automatically deletes the file when you exit the program and return to the Shell. However, unless you include the CALL command when you run a batch file from the Shell, or if you turn off your machine or reboot with the Shell loaded, DOS will leave the "temporary" file behind.

When you double-click on the GORILLA.BAS file, for example, DOS creates a temporary batch file that loads the Gorilla game using the command

```
@QBASIC \run C:\DOS\GORILLA.BAS
```

The reason these files end up in your \DOS directory is that when you install DOS 5, its installation program automatically includes the SET command

```
SET TEMP=C:\DOS
```

in your AUTOEXEC.BAT file. This command tells DOS to store in the \DOS directory the temporary files some programs create. If you've modified this command in your AUTOEXEC.BAT since installing DOS 5, you might find some of these batch files in whatever directory you specified.

Deleting the files

Once in a while, it's a good idea to get rid of these temporary files (and any temporary files created by your application programs). When you do, it's better to delete them from the prompt when you don't have any programs (including the Shell) loaded. That way, you won't run the risk of deleting a file that's currently in use.

One good way to do this is to change the TEMP setting to a directory other than \DOS. For instance, you could create a directory called \TEMP, then change the setting in your AUTOEXEC.BAT file to

```
SET TEMP=C:\TEMP
```

By doing that, you tell DOS to store all the temporary files in the \TEMP directory. Then, you could periodically go to the \TEMP directory and delete all the temporary files with the command

```
C:\TEMP>del *.*
```

You could then automate the process of cleaning out the TEMP directory by including the command

```
echo y | del %TEMP%\*.*
```

in your AUTOEXEC.BAT file. That way, DOS would automatically delete all the files in the \TEMP directory every time you turned on or rebooted your machine.

This command echoes the *Y* character to the DEL command's verification prompt. For an in-depth discussion of how this command works, see the letter "A Better Way to Avoid the DEL Command's Verification Prompt" in the March 1992 issue of *Inside DOS*.

We'd like to thank Legare Coleman for contributing some of the techniques described in this article.

ADVANCED BATCH FILE TECHNIQUE

VERSIONS 2.1-5.0

A command for updating files' date and time stamps

By Marco C. Mason

The project you're working on is finished, and it's time to put it on a disk and ship it out. But wait! All your files have different time and date stamps. You really want to change all the time and date stamps to the same value.

Why? If a customer or end user calls you with a problem, you want to be able to determine which version of your program he or she is running. If you keep a log of the date you created your software, then this job is much simpler.

Or, suppose you want to be able to do a simple backup, and you want to ensure that *all* the files in a certain directory get backed up. One way you can do this is by setting the current file date and time for all the files in the directory. Then, you can back up all the files from that date.

In either case, you need a way to set the time and date stamps. In this article, we'll show you a simple technique for doing just that. Then, we'll present a batch file you can use to update the date and time stamps of large groups of files.

What you need...

What you really need is a way to update the time and date stamps of all files. Surprisingly, it's pretty easy to do with DOS' COPY command. Simply type a command in the form

```
copy filename /b +
```

where *filename* is the name and extension of the file with the time and date stamp you want to update.

For example, to quickly update the date and time stamp of the PEDIGREE.TXT file, type

```
C:\>copy pedigree.txt /b +
```

When you press [Enter] and then issue the DIR command to check the date and time stamp in the directory listing, you'll see that DOS updated the file.

Why it works

Why does this command update the file's time and date stamp? Well, let's examine the plus symbol (+) first. You

use + to tell COPY you want to concatenate files. Typically, when you concatenate files, you list the files to concatenate and the name of the resulting file. If you leave off the name of the resulting file, the COPY command uses the first file specified.

Therefore, you're telling the COPY command to connect a NULL file to the end of the file you specify in *filename* and store the result in *filename*. This operation updates the date and time stamp of the file.

That's all well and good, but what's the /B switch for? The COPY command copies files in binary mode. In other words, it doesn't examine the contents of the file as it copies it. However, when you use COPY to concatenate files, it copies the files in text mode. Therefore, when it comes across the first end-of-file marker (^Z), it stops reading from the file. Additionally, when it has finished copying the file, it appends a ^Z to the file.

You don't want COPY to concatenate the files in text mode because if the file is a binary file, it might have ^Z embedded as a CPU instruction. If so, COPY would chop off the rest of the file. Additionally, you don't want COPY to add ^Z to the file. The /B switch tells COPY to treat the file as a binary file.

Building a batch file

Now that you know the easy way to update a file's time and date stamp, let's build a batch file, named UPDATE.BAT, that automates the process of updating *all* the files you specify. Figure A shows the batch file we created to update the time and date stamps of files.

Figure A

```
@echo off
:AGAIN
if %1x==x goto END
for %%i in (%1) do copy %%i /b +
shift
goto AGAIN
:END
```

UPDATE.BAT accepts multiple arguments, and updates the time and date stamp of every file you specify.

Let's take a quick look at how UPDATE.BAT works. The statement in line 3

```
if %1x==x goto END
```

checks to see if there's a file argument available—if not, the batch file ends. In other words, if you invoke UPDATE.BAT and fail to specify a filename by typing

```
C:\>update
```

the IF statement will route you to the label called :END, and return you to the system prompt.

If you do specify a filename, the statement

```
for %%i in (%1) do copy %%i /b +
```

carries out the COPY command on each file in the argument. The FOR command makes it possible for the batch file to operate on wildcard arguments. (For a complete discussion of the FOR command, see the article "Using the FOR Command to Issue Several Commands at Once" in the October 1990 issue of *Inside DOS*.)

Next, the SHIFT command throws away the first argument and shifts the second argument to the first, the third to the second and so on.

Finally, the command

```
goto AGAIN
```

returns you to the beginning of the batch file—the section labelled :AGAIN—which runs the batch file again. Eventually, you'll run out of arguments (DOS will copy the last file that meets your specification, and the IF statement in the third line will end the execution of the batch file).

Running the batch file

To run UPDATE.BAT, simply type *update*, followed by the filename or filespec you want to update. For example, to update all the files in the current directory whose extensions are TXT, type

```
C:\>update *.txt
```

When you press [Enter], DOS will copy all the files and update the date and time stamps of the TXT files. If you want to update all the files in the current directory, simply type

```
C:\>update *.*
```

Conclusion

Now you know the secret that major software companies use to make all the programs on their distribution disks have the same time and date. We've shown you how the COPY command works its magic, and, best of all, we've provided you with a simple batch file that you can use to update the date and time stamp of every file you specify.

Marco C. Mason is editor-in-chief of The Cobb Group's Inside Turbo C++ *and* Inside PC Tools *journals.*

Tooling along with DOS 5

Do you think of your software collection as a toolkit? Many software companies encourage us to think of their products as wrenches and screwdrivers, even the occasional Swiss Army knife. Although the tool metaphor is sometimes stretched thin, it's true that the measure of a program's worth is how well it does what it does; our favorite programs earn our loyalty because they let us work quickly and efficiently, not because they're written in C++ or represent the latest developments in OOP or GUI thinking.

DOS itself best fits this toolkit analogy. Its two principal functions are to keep the computer working smoothly and to let us manage our disks and files. The toolkit is pretty well-equipped: DEBUG, batch files, DOS commands, the Editor—all are tools we can use to work on our computers, just as we use a socket set or torque wrench to work on our cars. Each new version of DOS has added some tools to the kit; version 5 lets us do things that were formerly the province of special-purpose utility programs, if we're willing to spend a bit of time learning how to use the tools and understand how they work together.

For example, when you change a file, many programs (such as a word processor or drawing program) save the previous version of the file by changing its extension to BAK. These BAK files accumulate, taking up disk space; occasionally, it's necessary to find and delete these BAK files. You can either use the Shell to delete them all with just a few menu selections and a bit of typing, or you can build a tool that automates the task completely, letting you delete all BAK files with a single command.

Deleting all BAK files with the Shell

When the Shell is running and displaying the File List, follow these steps to find and delete all your BAK files:

1. Pull down the Options menu and choose the File Display Options... command, then type **.bak* in the Name field and press [Enter]. Now the File List displays the names of only those files in the current directory whose extension is BAK.

2. To display the names of the BAK files on the entire disk, select All Files from the View menu.

3. Highlight all the files, press the [Del] key (or select Delete from the File menu), and confirm the deletions.

That's it. Even on a large hard disk with dozens of directories and thousands of files, it takes just a couple of minutes. If there are some BAK files you don't want to delete, highlight just the ones you want to delete before you press the [Del] key.

Deleting all BAK files with one batch file

If you don't use the Shell, or would rather automate the process even further, you can combine the features of several DOS commands and programs to delete all the BAK files on your hard disk with a single batch file named UNBAK.BAT. This batch file uses the DIR command and Edlin to create a temporary batch file named $TEMP$.BAT, runs $TEMP$.BAT to delete the BAK files, then deletes $TEMP$.BAT. (Yes, that's right, Edlin: Don't worry, you won't have to use Edlin yourself.)

To build a batch file that deletes all the BAK files on your hard disk, you obviously have to start with the name of each BAK file. You can redirect the output of the DIR command, but how can you find *all* the BAK files?

Searching the whole disk with DIR

Version 5 makes the DIR command more versatile by adding the /S (*subdirectory*) switch which tells DIR to look for files in all subdirectories of the specified directory. Because the root directory is the upper-most directory of a disk, you can search the entire disk by specifying the root directory and the /S switch.

(The following examples assume that your hard disk is drive C and, to limit the space required, that you have three BAK files on your hard disk, in directories named WORD and CORELDRW. The technique works no matter how many BAK files you have.)

In order to display the names of all the files on your hard disk whose extension is BAK, enter the following command:

```
C:\>dir \*.bak /s
```

DOS responds by displaying the directory entry for each file whose extension is BAK. Subdirectories are identified by lines that start "Directory of", and summary lines show the number and size of all files found in each subdirectory and the number and total size of all files found:

```
Volume in drive C is RUBICON_ONE

Directory of C:\WORD

ANDY     BAK      4468 04-26-92  10:15p
           1 file(s)       4468 bytes

Directory of C:\CORELDRW

BRAND    BAK     68517 04-26-92  10:15p
MAP01    BAK    101322 04-26-92  10:15p
           2 file(s)     169839 bytes

Total files listed:
           3 file(s)     174307 bytes
                       87368704 bytes free
```

This report shows you where all the BAK files are and tells you how much disk space you can reclaim, but it wouldn't be much help in building a batch file, at least not with the tools that DOS provides. You have other tools, however, that produce a more usable result.

The bare necessities

Version 5 added another parameter to the DIR command, /B (for *bare*), which displays only the name and extension of each file; as an added feature, when you specify both the /S and /B switches, the DIR command adds the pathname to each filename instead of showing the directory name on a separate line. Compare the output of the following command with the previous example:

```
C:\>dir \*.bak /s /b

C:\WORD\ANDY.BAK
C:\CORELDRW\BRAND.BAK
C:\CORELDRW\MAP01.BAK
```

That's more like it. If you redirected the output of this command to a file named $TEMP$.BAT, all you'd have to do to make it a batch file that would delete all the BAK files on your hard disk is add the characters *del* followed by a space at the beginning of each line. You could edit the file with a word processor; multiple inserts from the scrap or clipboard would make it a fairly quick job. But another DOS tool lets you automate such mechanical editing: much-maligned Edlin, the text editor that's been around since year 1 (well, since version 1).

Hands-free editing

You may have seen the technique of redirecting the DEBUG program's input to a file of DEBUG commands in order to create a short assembly language program. This file of commands is often called a *script* file because once you start DEBUG, it follows the commands in the file instead of letting you type commands.

You can redirect Edlin's input to a script file, too. If you know that you'll always need the same commands, you can edit a file automatically by running Edlin from a batch file, redirecting Edlin's input to the script file. Adding the same four characters (in this case *del* and one space) to the beginning of each line in a file (which turns the redirected output of the DIR command into a batch file) is a perfect candidate for this technique.

An Edlin script file

Each line in $TEMP$.BAT starts with *C:*, and *C:* doesn't appear anywhere else in a line, so you can add *del* followed by a space at the beginning of each line by replacing *C:* with *del C:*. The Edlin REPLACE command, which adds *del* and a space to the beginning of each line, looks like this:

```
1,#rC:[Ctrl]zdel C:
```

The command is made up of five parts:

1,#	specifies the range of the command as the entire file (line 1 to the last line, represented by #);
r	specifies the REPLACE command;
C:	specifies the existing characters to be replaced;
[Ctrl]z	marks the end of the characters to be replaced;
del C:	specifies the new characters.

In addition to the REPLACE command, the Edlin script file must also include an Edlin EXIT command to return control to DOS. If you don't include the EXIT command, Edlin will sit forever waiting for more input from the script file and you'll have to restart your system.

Putting it all together

First, create the script file named UNBAK.SCR using the DOS Editor (or another text editor or word processor that lets you save a file without format codes). The file contains two lines:

```
1,#rC:→del C:
e
```

Most programs display [Ctrl]z as a right arrow character. To enter the → character using the DOS Editor, type [Ctrl]p, then [Ctrl]z. To enter the → character using Microsoft Word, hold down the [Alt] key, type 26 on the numeric keypad, and release the [Alt] key. And don't

forget that final line that contains nothing but *e*. Save the script file as UNBAK.SCR. It should be 18 bytes long.

Now create UNBAK.BAT, shown in Figure A. UNBAK.BAT creates, runs, and deletes $TEMP$.BAT.

Figure A

```
@echo off
dir c:\*.bak /s /b > $temp$.bat
edlin $temp$.bat < unbak.scr
call $temp$
del $temp$.*
```

UNBAK.BAT, together with UNBAK.SCR, automates the process of ridding your hard drive of BAK files.

Nothing fancy here:

- The first line turns off Echo.
- The second line creates $TEMP$.BAT by redirecting the output of a DIR command that finds all the BAK files on the disk.
- The third line edits $TEMP$.BAT by running Edlin and redirecting its input to UNBAK.SCR, the first file you created.
- The fourth line runs $TEMP$.BAT, deleting all the BAK files on the disk.
- The last line deletes $TEMP$.BAT (and $TEMP$.BAK, which Edlin created when it edited $TEMP$.BAT).

After making sure that the file matches the above lines, save it as UNBAK.BAT. It should be 101 bytes long.

Testing UNBAK.BAT

To test the batch file, first check how many BAK files are on your disk by typing

```
C:\>dir \*.bak /s /b
```

Check the list to make sure you want to delete all of them. If there are some you want to keep, use the ATTRIB command to make them read-only (type *attrib +r* and the filename). When you're ready to delete all your BAK files, type

```
C:\>unbak
```

If all goes well, you should see something like this:

```
End of input file
*1,#rC:^Zdel C:
          1:*del C:\WORD\ANDY.BAK
          2: del C:\CORELDRW\BRAND.BAK
          3: del C:\CORELDRW\MAP01.BAK
   *e
   C:>_
```

The filenames will be different, of course, but the last line before the prompt should be **e*. If DOS responds this way, make sure that it deleted the BAK files by typing

```
C:\>dir \*.bak /s /b
```

again; this time DOS should simply display the prompt without listing any filenames.

If DOS doesn't respond as shown and won't respond when you try to type something, you may have forgotten the second line of UNBAK.SCR that contains *e*. Restart your computer and proofread both UNBAK.SCR and UNBAK.BAT to make sure they're just as shown here, then test them again.

A cosmetic fix

You can, if you like, eliminate all response to UNBAK.BAT by redirecting Edlin's output to the NUL device (which

Creating UNBAK.SCR

In the article "Tooling Along with DOS 5" beginning on page 5, Van Wolverton creates a handy batch file called UNBAK.BAT. The heart of this batch file is an Edlin script called UNBAK.SCR. The Edlin script file is easy to create using the DOS 5 Editor—in fact it contains only two lines. However, it uses the special character →. Unlike all the other text in both UNBAK.BAT and UNBAK.SCR, you don't create the → symbol by pressing the → key on your keyboard.

To create UNBAK.SCR using the DOS Editor, type

```
C:\>edit unbak.scr
```

and press [Enter]. When you do, DOS will load the Editor and create the text file UNBAK.SCR. Next, type the following sequence of characters:

```
1,#rC:[Ctrl]p[Ctrl]zdel C:
e
```

Your screen will look like this:

```
1,#rC:→del C:
e
```

The combination of [Ctrl]p and [Ctrl]z creates the → character that UNBAK.SCR needs. In a future issue of *Inside DOS*, we'll take a more in-depth look at using special characters in script files.

just makes the output disappear). Add the redirection symbol and *nul* to the third line of UNBAK.BAT so that the line reads as follows:

```
edlin $temp$.bat < unbak.scr > nul
```

Now, when you type *unbak*, DOS deletes all the BAK files on your hard disk and displays the prompt again.

Although you may not need to delete all BAK files from your hard disk, these techniques, which create files by redirecting command output, edit files by redirecting Edlin's input to a script file, and create a batch file with another batch file, can be adapted to many other tasks. If you think of DOS as a kit of tools that work together, you'll find many ways to combine these tools to meet your needs.

Contributing editor Van Wolverton is the author of the best-selling books Running MS-DOS 5 *and* Supercharging MS-DOS. *Van, who has worked for IBM and Intel, currently lives in Alberton, Montana.*

COMMAND TECHNIQUES

VERSIONS 2.1-5.0

Take advantage of switches with SORT and FIND

The DOS FIND and SORT filter commands let you reorganize the output of other commands like DIR, TYPE, and FOR. Although the default settings for FIND and SORT are the ones you'll use most often, both commands have switches that let you further customize their actions. Learning to use these switches effectively will give you more options for using FIND and SORT.

In this article, we'll take a look at the switches for SORT and FIND and provide an example of a task requiring each of them. Since the FIND and SORT filters work with redirection characters like the pipe, we'll also look at ways to combine these commands to perform more than one operation with a single command. First, let's find out how to use switches.

Using command switches

Many DOS commands other than FIND and SORT have switches. FORMAT uses switches to designate the type of diskette or drive that it's going to use. BACKUP and RESTORE let you select which files to copy or replace. FIND and SORT use the same switch format as these other commands.

You enter DOS switches after the command by typing a forward slash (/) followed by the switch name. To get a directory list to pause after filling the screen, for instance, you type the command

```
C:\>dir /p
```

In certain circumstances, you may use more than one switch, such as

```
C:\>dir /p /w
```

This command shows your directory in a row-and-column layout and pauses after filling the screen.

Let's take a look at the switches DOS provides with SORT and FIND. Table A shows a quick reference guide for SORT and FIND switches.

Table A

SORT Switches	
/R (**R**everse order)	SORT Z to A
/+N (column **N**umber)	Begins SORT at column N
FIND Switches	
/C (**C**ounts lines)	Returns number of lines selected
/I (**I**gnore case)	Non-case sensitive FIND
/N (line **N**umbers)	Adds line numbers
/V (re**V**erse FIND)	Selects lines not matching criteria

FIND and SORT switches give you many options.

Using the SORT switches

The SORT command has two switches: /R and /+N. We created a text file called PEDIGREE.TXT containing the names of several family members. To begin, let's issue the SORT command without any switches:

```
C:\>type pedigree.txt | sort
```

DOS returns the following list:

```
Hey Bob Uhrebob
Jim Bob Smith
Jim Bob Smithfield
Joe Bob McCoy
Joe Bob Nesmith
Ray Bob Alperson
Ray Bob McCoy
Ted Bob Smith
```

DOS alphabetizes the names, beginning with the first letter of each line. The /R switch reverses the sorting order. To use the /R switch, enter the command

```
C:\>type pedigree.txt | sort /r
```

With the /R switch, DOS returns this list with the names appearing in reverse order:

```
Ted Bob Smith
Ray Bob McCoy
Ray Bob Alperson
Joe Bob Nesmith
Joe Bob McCoy
Jim Bob Smithfield
Jim Bob Smith
Hey Bob Uhrebob
```

The /+N switch is a little more complicated. Imagine your screen display completely filled with zeros. The characters would appear to be arranged into rows and columns, with each line containing the same number of zeros. This is how SORT sees your DOS screen. DOS numbers these columns, beginning with the first column on the left side of your display.

SORT processes each line beginning with the character in the first column of each line. You can ask SORT to begin in a column other than column 1 by assigning that number after the /+ in the switch. The + character must appear even though SORT won't let you assign a negative value. You could, for instance, re-sort the text file we've been using, but this time sort it beginning with column 9 where all the last names start. Enter the following command:

```
C:\>type pedigree.txt | sort /+9
```

DOS sorts the data beginning with the last names, all of which start in column 9:

```
Ray Bob Alperson
Ray Bob McCoy
Joe Bob McCoy
Joe Bob Nesmith
Jim Bob Smith
Ted Bob Smith
Jim Bob Smithfield
Hey Bob Uhrebob
```

If you format your data consistently, this switch can be especially helpful.

SORT has a couple of limitations, however. You can't sort a file larger than 64K. Most of the time, however, you would want to do a large sort in a word processor or database, anyway. Also, SORT always performs an alphabetical sort, so the number 1130 appears before the number 99 in a sorted list.

Using the FIND switches

As with SORT, the FIND command also includes switches. The FIND switches are /V, /C, /N, and /I. Two of these switches, /V and /I, work exclusively with the text of your document. The /V switch displays all lines which do not include the word you specify. The /I switch ignores case when searching for a word.

The other switches produce output related to the characteristics of the search FIND performs. The/C switch gives you the number of lines containing the word you specify. The final switch, /N, begins each line of the output list with the line number on which that reference appears.

First, let's look at /V. Continuing with the file PEDIGREE.TXT, type in the command

```
C:\>type pedigree.txt | find "Jim Bob" /v
```

DOS will respond

```
Ray Bob McCoy
Pay Bob Alperson
Joe Bob McCoy
Ted Bob Smith
Hey Bob Uhrebob
Joe Bob Nesmith
```

The list appears with all lines except the ones containing the name *Jim Bob*.

Now let's look at /I. The /I switch tells FIND to ignore case when searching for a word. For instance, if you asked DOS to find *jim bob* without using the /I switch, it would show no lines containing that name. The command

```
C:\>type pedigree.txt | find "ray bob" /i
```

returns the names *Ray Bob McCoy* and *Ray Bob Alperson*. While /I and /V work mainly with text, /N and /C produce numeric information. We'll look at /N next.

When you want to find out if a large text file contains the information you want, you can use FIND to see if it contains particular words or phrases. Knowing that a word is somewhere in a 100 screen document, however, does not help very much. The /N switch not only shows you which lines contain a particular word or phrase, but it includes the line numbers of the lines in the list.

While DOS doesn't have a command that lets you go to a specific line number within a document, by knowing the line number you can get a better sense of where you need to look for the information. The /N switch works just like the other switches. The command

```
C:\>type pedigree.txt | find "McCoy" /n
```

returns this list:

```
[3]Ray Bob McCoy
[5]Joe Bob McCoy
```

Finally, let's take a look at the /C switch. The /C switch gives you the number of lines containing a particular word or phrase. Unlike the other FIND switches, DOS simply returns a number when you use the /C switch. You need to remember that the /C switch gives you the number of lines containing a particular reference, and not the number of times that reference occurs in the document. If a line contains the word you specify in two places, DOS counts it only once.

You have to take the same precautions with FIND as you do with most utilities that search for a particular set of characters. If you enter

```
C:\>type pedigree.txt | find "SMITH"/i
```

DOS will give you the list

```
Jim Bob Smith
Jim Bob Smithfield
Ted Bob Smith
Joe Bob Nesmith
```

This list contains all names of which *smith* is a part. You can avoid this problem by putting a blank space before or after the word you want to find.

Combining switches

As we mentioned before, you can combine some of these switches. However, you must be careful not to give DOS conflicting instructions, specifically using the /C switch with any of the others. If you do, /C tries to return a number while the other switches want to generate a list. DOS cannot process both requests.

Some combinations are very useful, though. For instance, you can combine the /N and /I switches. This combination searches for a particular word and gives you a list of the numbered lines containing that word. Using the /I switch eliminates the need to enter all uppercase and lowercase combinations of the word you want to find.

You could combine the /+N and /R options in SORT to organize the PEDIGREE.TXT file in reverse alphabetical order sorted against the last names (which begin in column 9). The command is

```
C:\>type pedigree.txt | sort /r /+9
```

Take a few minutes and experiment with different switch combinations. This month's feature article, "Increase Your DOS Command-line Power with Pipe Filters," explains how to combine the SORT and FIND commands in a single statement.

Conclusion

You can increase the usefulness of SORT and FIND by using their option switches. The switches available with SORT and FIND give them a fuller range of functions. By combining switches in a single command, you can further increase the command's power.

Continued from page 1

Increase your DOS command-line power with...

A simple solution to a common problem

If you are like most DOS users, you've issued the command

```
C:\DOS>type readme.txt
```

only to see the desired section of a text file disappear from your screen before you could press [Pause]. For example, the README.TXT file included with DOS 5 provides a lot of useful information not covered in the DOS manual. You can look at the file (which is stored in the \DOS directory) by using the TYPE command. Although DIR provides the /P switch to pause the text after each screen, TYPE does not.

Fortunately, DOS supplies the MORE command. MORE displays information one screen at a time. After your screen fills with data, DOS pauses and lets you know that there's still some information to read. It does this by displaying the MORE indicator

```
--More--_
```

in the bottom left portion of your screen. You can press any key to see the remaining screens of data.

You may have tried using the MORE command with the < redirection symbol to present the text file in a page-by-page format. The command for this is

```
C:\DOS>more < readme.txt
```

The pipe symbol lets you use the MORE filter in the same way as you use the /P switch. The following command also produces page-by-page output but is a bit more intuitive if you're used to working with switches:

```
C:\DOS>type readme.txt | more
```

As you can see, the pipe takes the output from the TYPE command and processes it with the MORE filter. Although MORE has only one function and no switches, the other filters have considerable flexibility. Let's take a look at the other filters available with the pipe.

Filtering output with SORT and FIND

In most cases when you need to process large text files, you'll be using an application like a word processor or database program. At times, though, you'll want to search through a DOS text file for a certain piece of information. At other times, you may need to reorganize a DOS text file in a specific order. The SORT and FIND filters let you do this from the DOS command line.

SORTing command output

Although it has a few limitations, SORT offers you an easy way to alphabetize data from the DOS command line. Like MORE, SORT takes the output of a DOS command and reorganizes it. Most often, you'll use SORT with the pipe and the TYPE command.

We'll illustrate the SORT filter with the short text file called COMPANY.TXT shown in Figure A. This file contains the names of several businesses. We want to organize this information in several ways. You'll use the pipe symbol with the SORT and FIND commands to do this from the DOS command line.

Figure A

```
Kentucky Board of Education
Commonwealth Insurance of Kentucky
Allied Chemicals
Boscoe's Big Pins Bowling
Annie's Antiques
Zippy's Pets
Alfred Packer Memorial Gallery
First Kentucky Savings and Loan
```

COMPANY.TXT contains the names of several businesses.

The first task is to alphabetize the information according to the first word in each line. To do this, you use the SORT filter with the TYPE command. At the DOS prompt, type the command

```
C:\>type company.txt | sort
```

DOS sorts the file and produces the output shown below:

```
Alfred Packer Memorial Gallery
Allied Chemicals
Annie's Antiques
Boscoe's Big Pins Bowling
Commonwealth Insurance of Kentucky
First Kentucky Savings and Loan
Kentucky Board of Education
Zippy's Pets
```

DOS sorts the information into alphabetical order beginning with the leftmost character in each line.

FINDing references in a large text file

In many cases you won't need all the data in a text file. FIND lets you select only certain lines from a large text file by requesting that DOS give you only lines containing a specific word or phrase. Like SORT and MORE, FIND uses the TYPE command followed by the pipe character. With FIND, you must tell DOS what word to search for in the text file. To find a particular text string, use the following format:

```
type FILENAME.TXT | find "string"
```

where *FILENAME.TXT* is the name of the file you want to search and *string* is the name of the text string you want to locate.

You must follow the FIND command with a character string enclosed in quotation marks. Otherwise, you'll get an error message.

Unmodified, FIND is case sensitive, so be sure to enter the text string correctly. You've got to match capitals and lowercase letters exactly.

Now let's pull out only the companies with *Kentucky* in their names. You'll use the TYPE command again, only this time use the FIND filter. From the DOS command line, enter the following command:

```
C:\>type company.txt | find "Kentucky"
```

Look at the output file DOS sends to your screen. Notice that only the lines containing the word *Kentucky* appear. The lines, however, are still in the order in which you first entered them:

```
Kentucky Board of Education
Commonwealth Insurance of Kentucky
First Kentucky Savings and Loan
```

SORT and FIND are each powerful commands by themselves. It would be helpful, though, to find the lines you need as well as to sort them in alphabetical order in

a single command. Fortunately, DOS lets you use multiple filters with the pipe.

Using SORT and FIND together

Although you can accomplish either of the above tasks using the < redirection symbol, to sort and find information with the same command, you must use the pipe. The pipe allows you to send the same information through two different filters. Continuing with the same text file, let's get an alphabetical list of all the lines containing the word *Kentucky*.

From the DOS command line, enter the following command:

```
C:\>type company.txt | find "Kentucky" | sort
```

The command produces the same list of names as you generated with FIND by itself, except now the SORT command alphabetizes the list by the first word in each line:

```
Commonwealth Insurance of Kentucky
First Kentucky Savings and Loan
Kentucky Board of Education
```

The command sequence is important when using multiple filters. In the command you just issued, DOS locates all the records containing *Kentucky* and then sorts that list. Had you reversed FIND and SORT in the command, DOS would have sorted the entire list before selecting the records you wanted. The end result would have been the same, but the process would have taken longer.

Limitations and precautions

Although the pipe filters and redirection symbols offer you great flexibility, there are a few significant limitations. The most obvious limitation involves the SORT command. You can't sort a file larger than 64K. DOS will give you an *Out of memory* error. Most of the time, however, you will use an application program to sort files of this size.

Also, SORT can organize data only line-by-line and only beginning from a specified column position. You can't, for instance, alphabetize data based on the second or third word in each line.

The limitation you're most likely to encounter is the way SORT deals with numbers. SORT does not distinguish between numbers and letters. For this reason, DOS would sort the numbers 132, 33, 8, and 09 as follows:

```
09
132
33
8
```

If you have to work with numbers, be sure to include initial zeros so that all numbers have the same character length. The numbers 008, 009, 033, and 132 will sort correctly.

FIND has no provisions for wildcards (such as * or ?). You must match the string exactly, or FIND will not locate it. FIND won't search for phrases broken by a carriage return. If the whole phrase isn't on one line, FIND ignores it.

If you need to locate a phrase with quotation marks in it, you need to use double quotation marks ("") to indicate each quotation mark within the phrase. For example, to find the phrase *It was a "cute" bunny rabbit* you'd type the following command:

```
C:\> type FILENAME.TXT | find "It was a ""cute"" bunny rabbit"
```

Also, FIND will not locate a phrase broken by a carriage return so that part of the phrase is on one line and part of it is on another. For this reason, try to search for short phrases or single words whenever possible.

Conclusion

The pipe and other redirection symbols add much power and flexibility to your DOS command line. You can manipulate the output generated by DOS commands by using the MORE, FIND, and SORT filters.

DOS supplies both FIND and SORT with a number of switches. Using these switches allows you to customize their output in a number of ways. See "Take Advantage of Switches with SORT and FIND" on page 8 of this issue for more information.